Aspect-Oriented Crystal Ball

Adrian_Colyer@uk.ibm.com

The Rise of AO

"If I polish off my crystal ball, I foresee the rise of aspect-oriented programming"

- Grady Booch (IBM Fellow), 2003

Objectives

- Give you a sense of where AO is now, and where it's heading
  - in tools and languages
  - in the marketplace
  - in the research community
- ***Share some of IBM's thoughts and plans as all this unfolds…***
  - ***please contact the author for information on this part of the presentation***

Agenda

- Aspect-oriented languages
- Tool support
- Adoption of AO in marketplace
- Beyond Code
- ***IBM activities***
  - ***not included in this version***

Aspect Oriented Languages

What makes something AO?

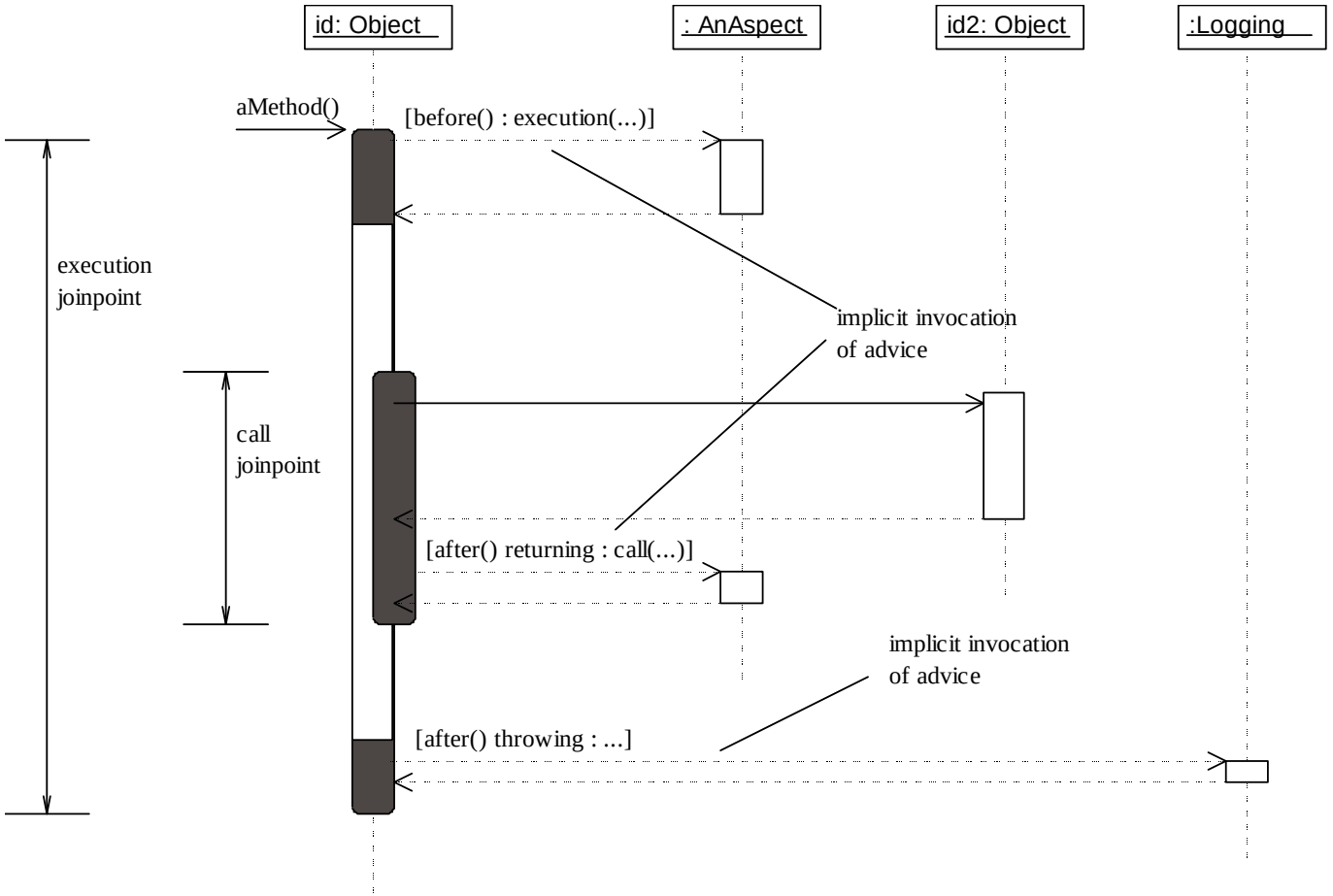
Some characteristics…

1. A richer set of referencing forms
   (- than what?)
   - “quantification”
   - composition and abstraction
2. Implicit invocation
   - related to “obliviousness”
3. Implicit instantiation
4. A unit of modularity for these new constructs
   - information hiding, abstraction

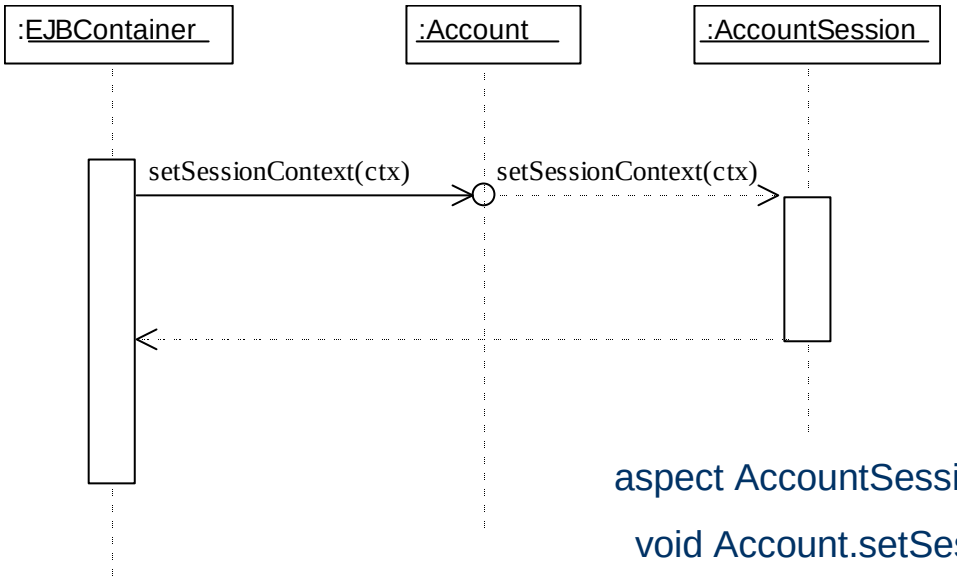
Reification in AspectJ…

- Referencing forms -> pointcut language
  - **pointcut** persistentFieldUpdate() :
    set(!transient * *);
- Implicit invocation
  - **after() returning** : persistentFieldUpdate() {…}
- Implicit instantiation
  - aspect Persistence **perthis**(persistentFieldUpdate()) {}
- Unit of modularity
  - **aspect** Persistence … { … }

Implicit Invocation - Advice



Implicit Invocation – Inter-type Declarations




```
aspect AccountSession {
  void Account.setSessionContext(Context ctx) {
  …
  }
}
```

Referencing forms

- Very rich set in natural language
  - “her, which, that,..” (pronouns)
  - “the president, the cat, the resident, and the hat” (nouns)
  - “colourless liquids” (constraint based)
  - “after reading the input stream” (temporal)
  - all combinations of the above
- Very limited set in programming languages
  - programs more complex than they need be

(Cristina Lopes)

Referencing forms

- AspectJ adds a richer set of dynamic, structural and temporal referencing forms
  - Pointcuts are a referencing form for a collection of join points
  - Advice specification is a temporal reference
- One direction is the addition of new referencing forms
  - e.g. structural form (concern/scope)

Declare scope…

```
declare scope : EJBSupport :
        types(com.ibm.ws.ejbcontainer..*) &&
        methods(* *ejb*(..);

pointcut ejbMethodCall() : call(* EJBSupport.*(..));
```

Expressing intent more directly

```
pointcut figureStateChange() :
   call(void FigureElement+.set*(..))  ||
   call(void FigureElement.moveBy(int, int));


after() returning : figureStateChange() {
  Display.update();
}
```

“If any displayed state changes, update the display…” ,

vs. “If methods matching a certain naming convention are called…”

[Gregor Kiczales]

Expressing intent more directly

```
declare scope : DisplayedState :
   pcflow(execution(void FigureElement+.draw()) &&
   get(* FigureElement+.*);

after() returning : set(* DisplayedState.*) {
   Display.update();
}
```

the predicted control flow of draw

fields accessed in that flow

Others: dflow(<pointcut>,<variable>)

- did the variable value come in a data flow through <pointcut> ?

AO Platform

- Development of aspect-oriented libraries
- Aspect-Oriented virtual machine?
  - preserve encapsulation in binary form
  - group dispatch instructions
  - consistent JPM (source vs binary)
    - eg handler, final fields
- “Pure” AO language
  - AspectJ : Java is like C++ : C

AspectJ in 2004

- 1.2 release scheduled for March/April
  - load-time weaving
  - performance and scalability enhancements
  - documentation and samples for popular J2EE servers
  - pertype instantiation model?
  - richer structure model surfaced for IDEs

AspectJ in 2004

- “Tiger” release 4Q04
  - supporting J2SE 1.5
  - generics
    - how do they impact pointcuts, advice and inter-type declarations?
  - metadata
    - both declare attribute, and join point selection based on attribute presence / values

JSR 175 Example

```
@Session( ejbName="statelessSession")
public class TraderEJB implements SessionBean {

  @Remote(transaction = Required)
  public void buy(String stockSymbol, int shares) {
    ...
  }
}
```

JSR 175 and AspectJ

- Property-based pointcut expressions…

```
pointcut sessionActivation(String name) :
   execution(* ejbActivation()) &&  attr(@Session(ejbName=name));
```

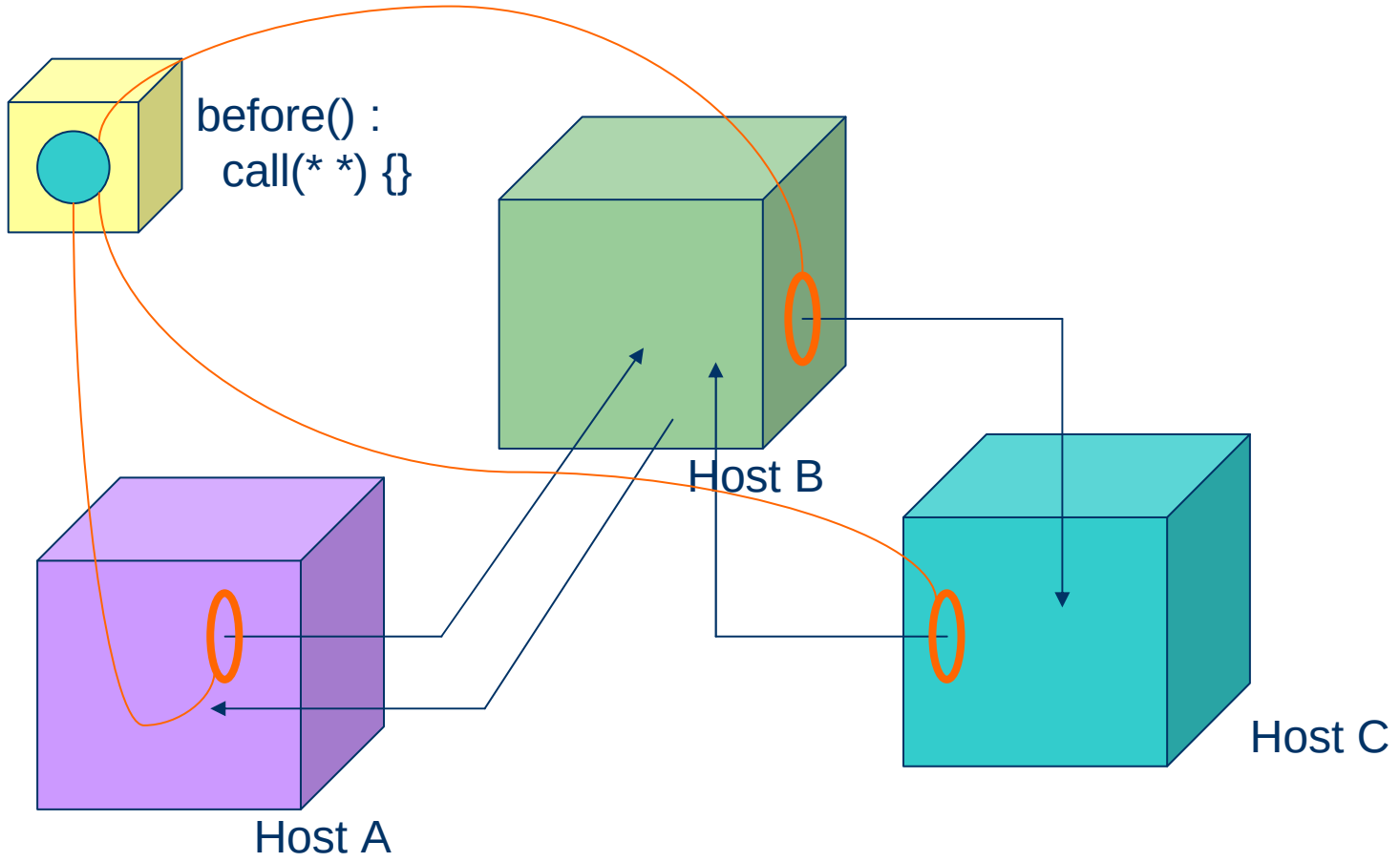
context binding?

has the Session attribute

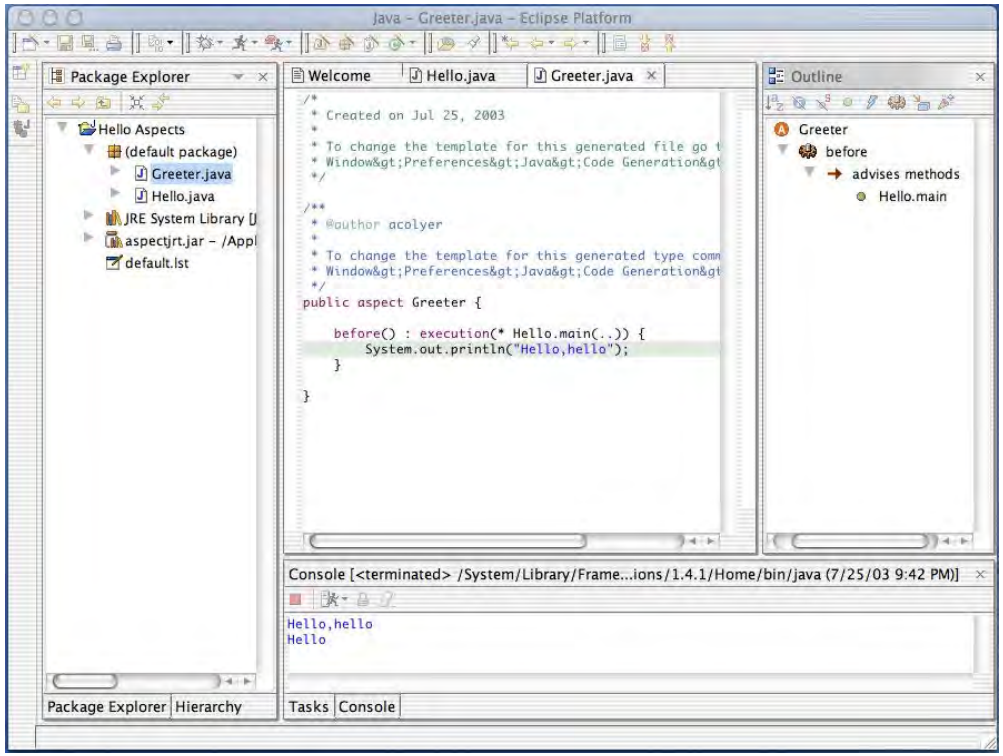
- Declare attribute…

```
declare attribute : TradeEJB hasattribute @Session(ejbName="name");
```

AspectJ in 2005?



Tools (IDE support)



IDE Support

- The bar is very high
  - excellent Java and C# tool support available
- Need to provide sufficiently rich AO experience
  - code completion
  - organize imports
  - multitude of browsers
  - fully incremental compilation
  - strong emphasis on showing and navigating relationships

Refactoring

- Rich support in the best IDEs
  - Eclipse, IntelliJ,…
- Examples:
  - rename
  - move
  - inline
  - Extract Method
  - …

Refactoring & AO

- rename

  Money balance()

  – what if the item to be renamed is named in a pointcut?
    - pointcut accountOperations() :

      call(* balance()) || call(…

  – what if the new name falls within a pointcut?
    - pointcut accessors() : execution(* get*(..))

Refactoring & AO

- Change signature
  - public Money getBalance()
    -> private Money getBalance()
- property based pointcut matching….
  - execution(public * Account.*(..))

Semantic Preservation

- Refactorings are supposed to preserve application semantics

```
public class Account {
  public Money getBalance();          private Money getBalance();
}
aspect AccountAuditor {
  pointcut publicAccInterface() :     pointcut publicAccInterface() :
   execution(public * Account.*(..));    execution(public * Account.*(..)) ||
 before() : publicAccInterface() {…      execution(Money  Account.getBalance());
}
```

Semantic Preservation

- Not based on matching exactly the same set of join points
- A higher level of expression is needed
  - specify intent more precisely
  - refactoring should preserve the intent

Refactoring

- We need to update the existing OO catalogue
- We can add new aspect-oriented refactorings
  - Extract Advice
    - program slicing?
  - Move Field to Inter-type Declaration
  - Move Method to Inter-type Declaration

AJDT in 2004

- Currently undergoing major rewrite
  - full incremental support
  - richer editing experience
  - integration of AspectJ structure in as many views as possible
  - debugging enhancements
- Follow-on release will probably address refactoring

Adoption

Language or framework?

- Changing the programming language (even as an extension) can be an adoption barrier
- Others have looked at alternate techniques
  - Java framework + XML
    - (AspectWerkz, JBoss)
  - Java framework + XDoc
    - (AspectWerkz)
  - Java framework + runtime attributes
    - (AspectWerkz
  - Plain Java + composition schema
    - (Hyper/J)

Runtime attributes example

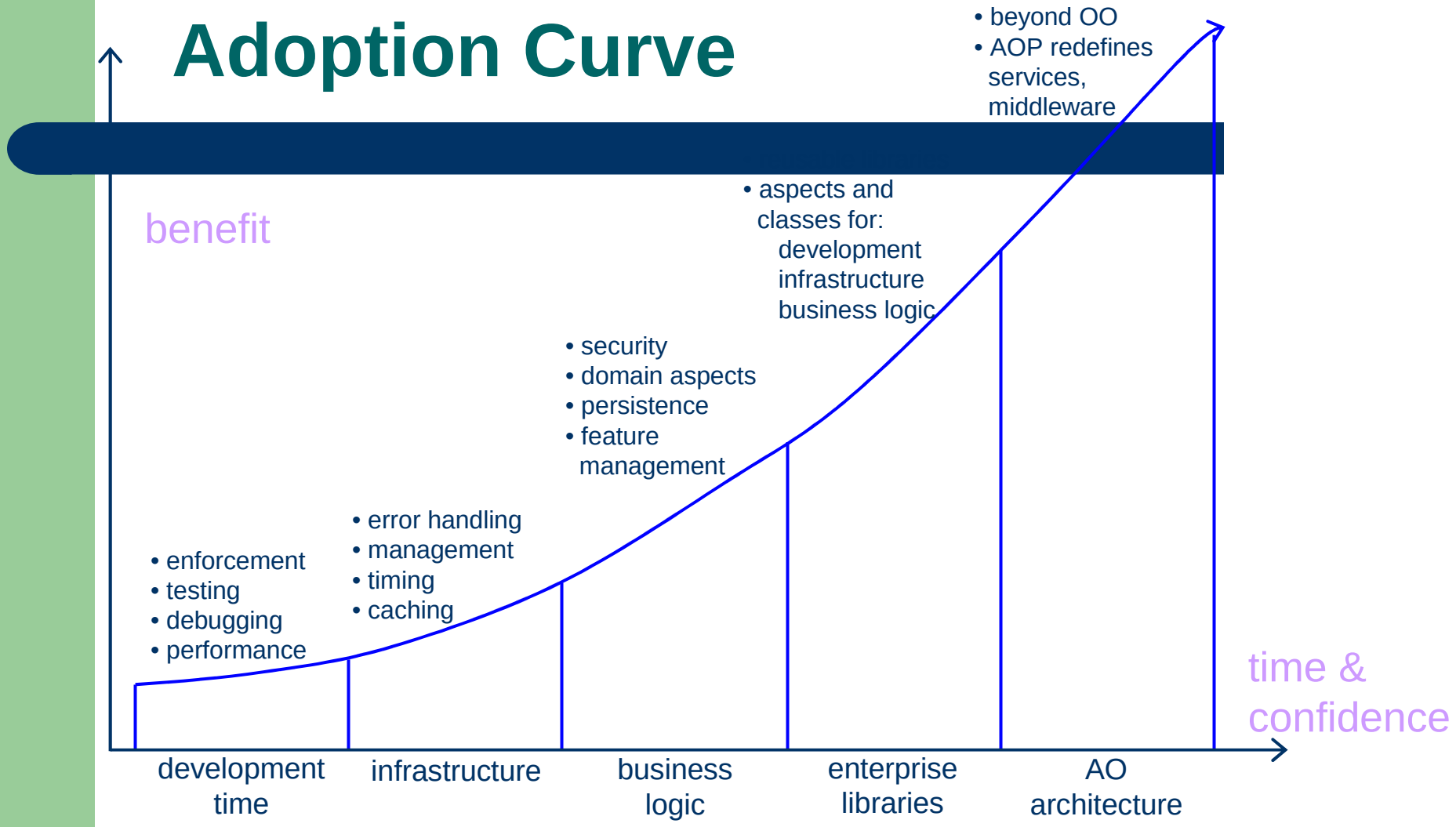
```
/**
 * @Aspect perInstance
 */
public class MyAspect extends Aspect {

  /**
   * @Call * foo.bar.*(..)
   */
  Pointcut methodsToLog;

  /**
   * @Before methodsToLog
   */
  public void logMethod(JoinPoint joinPoint) throws Throwable {
    // log method entry...
  }

}
```

[AspectWerkz]

Weave-time?

- Pre-compilation (source -> source)
  - AspectJ 1.0.x
- Compilation (source -> binary)
  - AspectJ
- Post-compilation (binary -> binary)
  - AspectJ, AspectWerkz, Hyper/J
- Load-time
  - AspectWerkz, JBoss, (AspectJ, Hyper/J)
- Runtime
  - AspectWerkz, JBoss (limited)

Adoption Curve



‘Real-World’ Adoption is Growing

“My company uses it [AspectJ] for a client desktop application that is deployed to over 3000 agents. We have had aspects in production for almost 2 years now and have not seen any problems, etc. with them. In fact part of the aspects are applied at in our data layer, which is our most critical part of the system. We have performance and load tested it to prove to ourselves and our customers that there is no more impact with aspects than with normal code.”

- Ron DiFrango, CapitalOne

‘Real-World’ Adoption is Growing

“Besides the few stumbling blocks that we've encountered, AspectJ is a great software development tool. With certain problems that crosscut the entire system the AOP way feels more natural than the copy and pasted look of traditional OO. If you don't feel completely ready to trust it, it can be introduced slowly. We started out that way but once we saw the benefits of using it we started using it everywhere.”

- Jason Gilman, ARINC

We're discovering the core aspects

- 'Tracing' is to AO, as 'String' is to OO
- General purpose applications:
  - tracing, logging, error handling, profiling,
  - caching, pooling,…
- Enterprise applications have a whole new set of well-known xcc's
  - transactions, persistence, security, distribution,…
  - couple with a need to reduce complexity and increase flexibility
  - an obvious match
- There are many, many more domain and application specific aspects.

Enterprise AO focus

- Use aspects to simplify business object development
  - POJO's plus aspects for required system services
- Load (&run) time instrumentation of applications
  - largely targeted at auxiliary aspects
- Open up containers for extension
  - add new services as aspects

BEA WebLogic Aspect Framework

- An ‘aspect’ library
- A mechanism for deploying aspects into the server
  - load-time weaving of user applications
- An SPI for integrating AO ‘weavers’ with the app server
  - AspectJ is the only one supplied out of the box
  - AspectWerkz strong second contender

JBoss AOP

- Own AOP framework
- Hot deployment
- Metadata and metadata chains
- Add/remove interceptors at runtime
- **Pre-supplied AOP services**
- AOP Management console
  - see interceptor chains, introductions, metadata

JBoss AOP Services

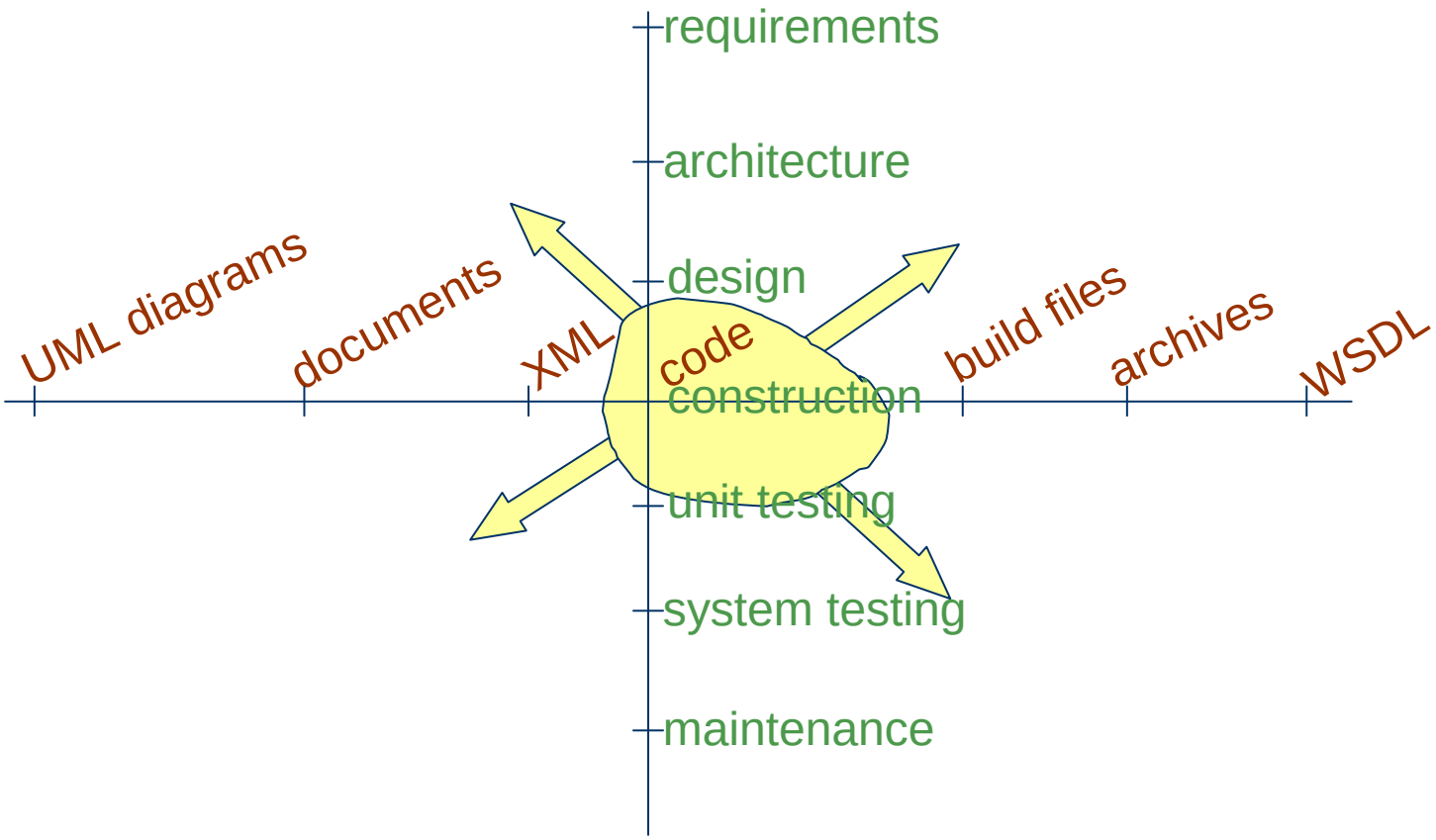
- Transactions
- Security
- Remoting, Clustered Remoting
- Transactional locking
- Transactional cache, clustered tx cache
  - object versioning

JBoss AOP

- Aspects applied at loadtime
  - 10-15% overhead on class load
- Can only hot-deploy aspect features onto classes advised when first loaded, or explicitly marked as advisable
- limited pointcut language

Beyond Code…

AO Space


requirements
architecture
design
code
construction
unit testing
system testing
maintenance
UML diagrams
documents
XML
build files
archives
WSDL

Design Patterns

- A library of problems and (OO) solutions
- Does AO offer alternative (better in some way) solutions?
  - Yes (Hanneman & Kiczales, 2002)
  - 17/23 GoF patterns have improved modularity with AspectJ
    - 12/17 permit core part of pattern to be captured in abstract aspect
    - 14/17 allow transparent composition of pattern instances

Design Patterns (Library aspect)

```
public abstract aspect ObserverProtocol perthis(observedBehaviour(Subject))
{
  protected interface Subject {};
  protected interface Observer {};

  private List observers = new LinkedList();

  public void addObserver(Observer o) {
    observers.add(o);
  }

  public void removeObserver(Observer o) {
    observers.remove(o);
  }
```

Design Patterns (Library aspect)

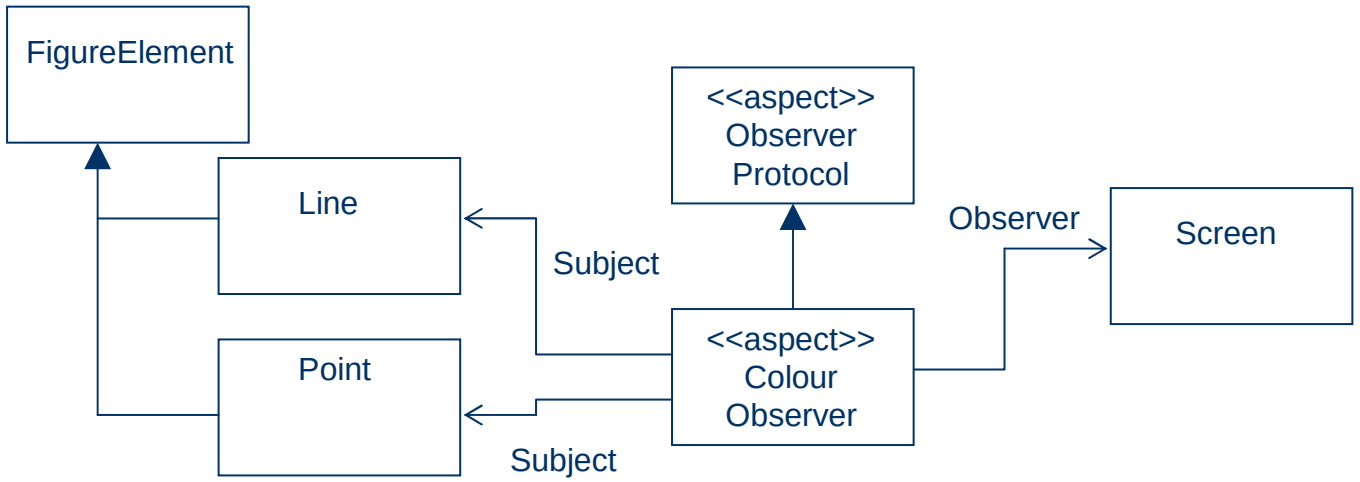
```
  abstract protected pointcut observedBehaviour(Subject s);

  abstract protected void updateObserver(Subject s, Observer o);

  after(Subject s) returning : observedBehaviour(s) {
    for (Iterater it = observers.iterator(); it.hasNext(); ) {
      updateObserver(s,(Observer)it.next());
    }
  }
}
```

Design Patterns

```
public aspect ColourObserver extends ObserverProtocol {

  declare parents : Point implements Subject;
  declare parents : Line implements Subject;
  declare parents : Screen implements Observer;

  protected pointcut observedBehaviour(Subject s) :
    execution(* Point.setColor(Color)) ||
    execution(* Line.setColor(Color)) &&
    this(s);

  protected void updateObserver(Subject s, Observer o) {
    ((Screen)o).updateColorOf((FigureElement)s);
  }
}
```

Design Patterns

- Modularised implementation
- Representation by association in UML
  - map observedBehaviour pointcut



AOSD in IBM

External Activities

- Leading the AspectJ project
  - http://www.eclipse.org/aspecj
- Leading the AspectJ Development Tools project
  - http://www.eclipse.org/ajdt
- Leading the CME project
  - http://www.eclipse.org/cme [soon]
- Very active in AOSD community

IBM Activities

- ***Other details on IBM's activities omitted from this version***

Summary

- Interesting developments in…
  - languages
  - tools
  - methodology
- Adoption is beginning in earnest
- IBM is taking a leading role